Analiza SOLID principa

**1. Single Responsibility Principle (SRP)**

*Klasa treba imati samo jedan razlog za promjenu, tj. treba imati jednu jasno definisanu odgovornost.*

U ovom sistemu SRP je ispoštovan na sljedeći način:

1. **Korisnik**: modeluje osnovne podatke o korisnicima sistema (ime, prezime, email itd.).
2. **Rezervacija**: upravlja podacima o rezervacijama (projekcija, korisnik, status, cijena).
3. **Karta**: sadrži informacije o dodijeljenim kartama i povezanim sjedištima.
4. **Cijena**: odgovorna je za izračun konačne cijene, koristeći osnovnu cijenu i odgovarajući popust.

**2. Open/Closed Principle (OCP)**

*Softverski entiteti treba da budu otvoreni za proširenje, ali zatvoreni za modifikaciju.*

Dizajn sistema podržava ovaj princip na više načina:

* Klasa **Korisnik** je apstraktna, a iz nje se nasljeđuju konkretni tipovi korisnika kao što su Administrator, RegistrovaniKorisnik i Radnik. Novi tipovi korisnika mogu se lako dodati bez potrebe da se mijenja osnovna klasa Korisnik.
* Interfejs **Popust** omogućava implementaciju raznih vrsta popusta (StudentskiPopust, RadničkiPopust, PenzionerskiPopust). Dodavanje nove vrste popusta (npr. „VikendPopust“) moguće je bez izmjene postojećih klasa.

**3. Liskov Substitution Principle (LSP)**

*Objekti izvedenih klasa treba da se mogu koristiti umjesto objekata osnovne klase, bez narušavanja funkcionalnosti.*

Klase Administrator, RegistrovaniKorisnik i Radnik su podklase apstraktne klase Korisnik i u svim kontekstima se mogu koristiti gdje se očekuje objekat tipa Korisnik. Tako se, npr., u klasi Rezervacija koristi identifikator korisnika (korisnik\_id) bez obzira na konkretan tip korisnika.

**4. Interface Segregation Principle (ISP)**

*Klijenti ne bi trebali biti prisiljeni da zavise od interfejsa koje ne koriste.*

Interfejs Popust sadrži samo jednu metodu: izracunajPopust(), koju sve klase popusta implementiraju. Ovim je postignuto da svaka konkretna implementacija (StudentskiPopust, RadničkiPopust, PenzionerskiPopust) koristi samo ono što joj je zaista potrebno, bez opterećenja dodatnim metodama.

**5. Dependency Inversion Principle (DIP)**

*Moduli visokog nivoa ne treba da zavise od modula niskog nivoa, već oba treba da zavise od apstrakcije.*

Ovaj princip je ispoštovan kroz odnos između klase Cijena i interfejsa Popust. Klasa Cijena zavisi isključivo od interfejsa Popust, a ne od konkretnih implementacija. Tako, logika za izračunavanje popusta može biti izmijenjena ili proširena (dodavanjem novih vrsta popusta) bez potrebe da se mijenja sama klasa Cijena.